```
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR   RRR       RRR   RRR             TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF               000         000 RRR     RRR     RRR     RRR           TTT        LLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR         RRR RRR         RRR       TTT        LLLLLLLLLLLLLLL
```

FILEID**FORUDFRF

```
FFFFFFFFFF    000000    RRRRRRRR    UU      UU  DDDDDDDD    FFFFFFFFFF  RRRRRRRR    FFFFFFFFFF
FFFFFFFFFF    000000    RRRRRRRR    UU      UU  DDDDDDDD    FFFFFFFFFF  RRRRRRRR    FFFFFFFFFF
FF          00      00  RR      RR  UU      UU  DD      DD  FF          RR      RR  FF
FF          00      00  RR      RR  UU      UU  DD      DD  FF          RR      RR  FF
FF          00      00  RR      RR  UU      UU  DD      DD  FF          RR      RR  FF
FF          00      00  RR      RR  UU      UU  DD      DD  FF          RR      RR  FF
FFFFFFFF    00      00  RRRRRRRR    UU      UU  DD      DD  FFFFFFFF    RRRRRRRR    FFFFFFFF
FFFFFFFF    00      00  RRRRRRRR    UU      UU  DD      DD  FFFFFFFF    RRRRRRRR    FFFFFFFF
FF          00      00  RR   RR     UU      UU  DD      DD  FF          RR   RR     FF
FF          00      00  RR   RR     UU      UU  DD      DD  FF          RR   RR     FF
FF          00      00  RR    RR    UU      UU  DD      DD  FF          RR    RR    FF          ....
FF          00      00  RR    RR    UU      UU  DD      DD  FF          RR    RR    FF          ....
FF            000000    RR      RR  UUUUUUUUUU  DDDDDDDD    FF          RR      RR  FF          ....
FF            000000    RR      RR  UUUUUUUUUU  DDDDDDDD    FF          RR      RR  FF          ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE FOR$$UDF_RF (%TITLE 'FORTRAN Read Formatted UDF'
;     2          0002  0                  IDENT = '1-043'          ! File: FORUDFRF.B32  Edit: SBL1043
;     3          0003  0                  ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                   *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 ! FACILITY:  FORTRAN Support Library - not user callable
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       This module implements FORTRAN Read Formatted I/O
;    35          0035  1 !       statements (sequential access - S, direct access - D,
;    36          0036  1 !       DECODE - M) at the User data Formatter level of
;    37          0037  1 !       abstraction (UDF level is 2nd level). This module
;    38          0038  1 !       calls the Read/write independent format
;    39          0039  1 !       interpreter (FOR$INTERP) to decode the compiled format
;    40          0040  1 !       statement. This module calls the appropriate read record
;    41          0041  1 !       routine at the record handling level of abstraction (REC
;    42          0042  1 !       level is 3rd level) to read a record.
;    43          0043  1 !
;    44          0044  1 ! ENVIRONMENT:  User access mode; reentrant AST level or not.
;    45          0045  1 !
;    46          0046  1 ! AUTHOR:  Thomas N. Hastings;   CREATION DATE: 20-Feb-77
;    47          0047  1 !
;    48          0048  1 ! MODIFIED BY:
;    49          0049  1 ! [Previous edit history removed.  SBL 29-Oct-1982]
;    50          0050  1 ! 1-036 - Instead of using zero ELEM_SIZE to determine a call from
;    51          0051  1 !         FOR$$UDF_RF9, use a zero ELEM_TYPE.  This allows
;    52          0052  1 !         zero-length strings to be processed correctly.
;    53          0053  1 !         SPR 11-30127  SBL 22-May-1980
;    54          0054  1 ! 1-037- Use new F_floating input conversion routine, OTS$CVT_T_F.
;    55          0055  1 !         JAW 14-Apr-1981
;    56          0056  1 ! 1-038 - Convert FOR$$FMT_INTRP1 to JSB linkage.  JAW 29-Jul-1981
;    57          0057  1 ! 1-039 - Use OTS$CVT_T_F instead of OTS$CVT_T_D when format is D/E/F/G
```

```
;    58          0058  1 !                 and element is not floating (FORVARMIS).  JAW 05-Aug-1981
;    59          0059  1 ! 1-040 - Add require file FORMSG.B32 in preparation for enhanced error
;    60          0060  1 !                 reporting.  JAW 10-Aug-1981
;    61          0061  1 ! 1-041 - Cite text in error and current record number when signaling
;    62          0062  1 !                 INPCONERR.  JAW 27-Aug-1981
;    63          0063  1 ! 1-042 - For indexed and internal files, use a secondary message that doesn't
;    64          0064  1 !                 put out a record number (INVTEX).  DGP  21-Dec-1981
;    65          0065  1 ! 1-043 - Change to use FORPROLOG.REQ.  Make references to OTS$CVT routines PIC.
;    66          0066  1 !                 SBL 29-Oct-1982
;    67          0067  1 !--
;    68          0068  1
```

```
;    70          0069  1 !
;    71          0070  1 ! PROLOGUE FILE:
;    72          0071  1 !
;    73          0072  1
;    74          0073  1 REQUIRE 'RTLIN:FORPROLOG';                      ! FOR$ definitions
;    75          0139  1 SWITCHES ZIP;                                  ! Optimize for speed
;    76          0140  1
;    77          0141  1 !
;    78          0142  1 ! TABLE OF CONTENTS:
;    79          0143  1 !
;    80          0144  1
;    81          0145  1 FORWARD ROUTINE
;    82          0146  1     FOR$$UDF_RF0 : JSB_UDF0 NOVALUE,           ! initialization
;    83          0147  1     FOR$$UDF_RF1 : CALL_CCB NOVALUE,           ! format one user I/O list element
;    84          0148  1     FOR$$UDF_RF9 : JSB_UDF9 NOVALUE,           ! end of user I/O list - finish
;    85          0149  1     DO_READ : JSB_DO_READ NOVALUE,             ! do per-record formatting and read
;    86          0150  1     MOVE_CHAR : NOVALUE,                       ! Same as CH$MOVE
;    87          0151  1     COPY_CHAR;                                 ! Same as CH$COPY
;    88          0152  1
;    89          0153  1 !
;    90          0154  1 ! MACROS:
;    91          0155  1 !
;    92          0156  1
;    93          0157  1 MACRO
;    94        M 0158  1     RF_EOLST =
;    95          0159  1  0,7,1,0%,                                     ! Check for end of user I/O list
;    96        M 0160  1     RF_CHECKW =
;    97          0161  1  0,6,1,0%,                                     ! Check for w positions left
;    98        M 0162  1     RF_SHORT =
;    99          0163  1  0,5,1,0%,                                     ! Check for short string
;   100          0164  1 !             0,4,1,0%      spare
;   101        M 0165  1     RF_DISPAT =
;   102          0166  1  0,0,4,0%;                                     ! CASE index for dispatch
;   103          0167  1
;   104          0168  1 MACRO                                          ! Attribute packing macro for attribute table
;   105        M 0169  1     A (E, W, S, NDX) =
;   106          0170  1  (E^7 + W^6 + S^5 + NDX)%;
;   107          0171  1
;   108          0172  1 !
;   109          0173  1 ! EQUATED SYMBOLS:
;   110          0174  1 !
;   111          0175  1 !    NONE
;   112          0176  1 !
;   113          0177  1 ! OWN STORAGE:
;   114          0178  1 !
;   115          0179  1
;   116          0180  1 BIND
;   117          0181  1     RF_ACT =                                   ! Action table for UDF_RF1, UDF_RF9 format codes
;   118          0182  1 !+
;   119          0183  1 ! The format codes are structured as follows:
;   120          0184  1 ! 0 - do nothing
;   121          0185  1 ! 1 - call intermediate record processing routine
;   122          0186  1 ! 2 - do nothing
;   123          0187  1 ! 3 - not used
;   124          0188  1 ! 4 - move right (old X format)
;   125          0189  1 ! 5 - copy Hollerith
;   126          0190  1 ! 6 - return no. of character positions remaining
```

```
;  127          0191  1 ! 7 - copy alpha strings
;  128          0192  1 ! 8 - all integer format processing
;  129          0193  1 ! 9 - all floating format
;  130          0194  1 !-
;  131          0195  1         UPLIT BYTE(
;  132          0196  1
;  133          0197  1 !         E C S   EOLST - End of I/O list
;  134          0198  1 !         O H H   CHECKW - Set up descriptor; check field width
;  135          0199  1 !         L E O   SHORT - Check for short input field
;  136          0200  1 !         S C R
;  137          0201  1 !         T K T
;  138          0202  1 !           W                dec     hex
;  139          0203  1           A(1,0,0, 0),    ! ER     = 0,    ! 00    ! format syntax error
;  140          0204  1           A(0,0,0, 0),    ! LP     = 1,    ! 01    ! ( - format reversion point
;  141          0205  1           A(0,0,0, 0),    ! NLP    = 2,    ! 02    ! n( - left paran of repeat group
;  142          0206  1           A(0,0,0, 0),    ! )      = 3,    ! 03    ! ) - right paren of repeat group
;  143          0207  1                           ! MAINTENANCE NOTE: the above should not be seen by this module,
;  144          0208  1                           ! except look ahead in FOR$$UDF_RF9
;  145          0209  1           A(1,0,0, 1),    ! EOF    = 4,    ! 04    ! ) - End of format
;  146          0210  1           A(0,0,0, 1),    ! SLS    = 5,    ! 05    ! / - Record separator
;  147          0211  1           A(0,0,0, 2),    ! DLR    = 6,    ! 06    ! $ - Dollar sign: terminal I/O
;  148          0212  1           A(1,0,0, 0),    ! CLN    = 7,    ! 07    ! : - Colon: terminate if end of list
;  149          0213  1           0,              ! UNUSED  8
;  150          0214  1           0,0,0,          ! Not seen here  9:11
;  151          0215  1           A(0,0,0, 0),    ! _P     = 12,   ! 0C    ! sP - signed scale factor
;  152          0216  1           A(0,0,0, 0),    ! _T     = 13,   ! 0D    ! Tn - Tab Set
;  153          0217  1                           ! Above code only seen by lookahead
;  154          0218  1           A(0,0,0, 4),    ! _X     = 14,   ! 0E    ! nX - Skip n columns
;  155          0219  1           A(0,1,0, 5),    ! _H     = 15,   ! 0F    ! nHcccc - Hollerith
;  156          0220  1           0,0,            ! Not seen here  16:17
;  157          0221  1           A(0,0,0, 0),    ! TL     = 18,   ! 12    ! TLn - Tab left n
;  158          0222  1           A(0,0,0, 0),    ! TR     = 19,   ! 13    ! TRn - Tab right n
;  159          0223  1                           ! Above two only seen by lookahead
;  160          0224  1           A(1,0,0, 6),    ! _Q     = 20,   ! 14    ! Q
;  161          0225  1           A(1,1,0, 7),    ! _A     = 21,   ! 15    ! nAw - Alpha numeric
;  162          0226  1           A(1,1,1, 8),    ! _L     = 22,   ! 16    ! nLw - Logical
;  163          0227  1           A(1,1,1, 8),    ! _O     = 23,   ! 17    ! nOw - Octal
;  164          0228  1           A(1,1,1, 8),    ! _I     = 24,   ! 18    ! nIw - Integer
;  165          0229  1           A(1,1,1, 8),    ! _Z     = 25,   ! 19    ! nZw - Hexadecimal
;  166          0230  1           A(1,1,1, 8),    ! XO     = 26,   ! 1A    ! Ow.m - Extended O
;  167          0231  1           A(1,1,1, 8),    ! XI     = 27,   ! 1B    ! Iw.m - Extended I
;  168          0232  1           A(1,1,1, 8),    ! XZ     = 28,   ! 1C    ! Zw.m - Extended Z
;  169          0233  1           0,              ! UNUSED  29
;  170          0234  1           A(1,1,1, 9),    ! _F     = 30,   ! 1E    ! nFw.d - Fixed format
;  171          0235  1           A(1,1,1, 9),    ! _E     = 31,   ! 1F    ! nEw.d - Scientific notation format
;  172          0236  1           A(1,1,1, 9),    ! _G     = 32,   ! 20    ! nGw.d - General format
;  173          0237  1           A(1,1,1, 9),    ! _D     = 33,   ! 21    ! nDw.d - Double Precision format
;  174          0238  1           A(1,1,1, 9),    ! XE     = 34,   ! 22    ! nEw.dEe
;  175          0239  1           A(1,1,1, 9),    ! XG     = 35,   ! 23    ! nGw.dEe
;  176          0240  1                           ! The following codes are used for lookahead only
;  177          0241  1           0,0,0,0,0,      ! UNUSED  36:40
;  178          0242  1           A(1,0,0, 0),    ! _DA    = 41    ! 29    ! nA - default A
;  179          0243  1           A(1,0,0, 0),    ! _DL    = 42    ! 2A    ! nL - default L
;  180          0244  1           A(1,0,0, 0),    ! _DO    = 43    ! 2B    ! nO - default O
;  181          0245  1           A(1,0,0, 0),    ! _DI    = 44    ! 2C    ! nI - default I
;  182          0246  1           A(1,0,0, 0),    ! _DZ    = 45    ! 2D    ! nZ - default Z
;  183          0247  1           0,0,0,0,        ! UNUSED  46:49
```

```
;  184     0248  1          A(1,0,0, 0),    ! _DF   = 50    ! 32    ! nF - default F
;  185     0249  1          A(1,0,0, 0),    ! _DE   = 51    ! 33    ! nE - default E
;  186     0250  1          A(1,0,0, 0),    ! _DG   = 52    ! 34    ! nG - default G
;  187     0251  2          A(1,0,0, 0)     ! _DD   = 53    ! 35    ! nD - default D
;  188     0252  1          ) : VECTOR [54, BYTE];
;  189     0253  1
;  190     0254  1  !+
;  191     0255  1  ! Declare table of conversion routine addresses.  This will be filled in
;  192     0256  1  ! by FOR$$UDF_RF0 upon first entry.  Entries 0-3 are the integer conversion
;  193     0257  1  ! routines for the formats L, O, I and Z, respectively.  The only other
;  194     0258  1  ! elements filled in are those corresponding to datatypes F, D, G and H;
;  195     0259  1  ! these elements are indexed by the DSC$K datatype code.
;  196     0260  1  !-
;  197     0261  1
;  198     0262  1  OWN
;  199     0263  1      AA_IN_CVT: VECTOR [DSC$K_DTYPE_H+1, LONG],
;  200     0264  1      CVT_INIT: INITIAL (0);                          ! 1 if array initialized
;  201     0265  1
;  202     0266  1  !
;  203     0267  1  ! EXTERNAL REFERENCES:
;  204     0268  1  !
;  205     0269  1
;  206     0270  1  EXTERNAL
;  207     0271  1      FOR$$AA_REC_PR0 : VECTOR,                       ! PIC array of record processor
;  208     0272  1                                                      ! procedure-initializations in REC
;  209     0273  1                                                      ! level of abstraction. Indexed by
;  210     0274  1                                                      ! I/O statement type (ISB$B_STTM_TYPE)
;  211     0275  1      FOR$$AA_REC_PR1 : VECTOR;                       ! PIC array of record processor procedures
;  212     0276  1
;  213     0277  1                                                      ! Read a record in REC level of
;  214     0278  1                                                      ! abstraction. Indexed by I/O statement
;  215     0279  1                                                      ! type (ISB$B_STTM_TYPE)
;  216     0280  1
;  217     0281  1  EXTERNAL ROUTINE
;  218     0282  1      OTS$CVT_T_F,                                    ! F-only input conversion
;  219     0283  1      OTS$CVT_T_D,                                    ! F and D input conversion
;  220     0284  1      OTS$CVT_T_G,                                    ! G input conversion
;  221     0285  1      OTS$CVT_T_H,                                    ! H input conversion
;  222     0286  1      OTS$CVT_TL_L,                                   ! L format input conversion
;  223     0287  1      OTS$CVT_TO_L,                                   ! O format input conversion
;  224     0288  1      OTS$CVT_TI_L,                                   ! I format input conversion
;  225     0289  1      OTS$CVT_TZ_L,                                   ! Z format input conversion
;  226     0290  1      FOR$$FMT_INTRP0 : JSB_FMT0 NOVALUE,             ! initialize format interpreter
;  227     0291  1      FOR$$FMT_INTRP1 : JSB_FMT1 NOVALUE,             ! get next data format code
;  228     0292  1                                                      ! or input-output format code
;  229     0293  1      FOR$$SIGNAL : NOVALUE,                          ! convert FORTRAN err # to
;  230     0294  1                                                      ! VAX error # and SIGNAL
;  231     0295  1      FOR$$SIGNAL_STO : NOVALUE;                      ! convert FORTRAN err # to
;  232     0296  1
;  233     0297  1                                                      ! VAX error # and SIGNAL_STOP
;  234     0298  1
;  235     0299  1
```

```
;  237     0300  1 GLOBAL ROUTINE FOR$$UDF_RF0                    ! Read formatted UDF initialization
;  238     0301  1     : JSB_UDF0 NOVALUE =
;  239     0302  1
;  240     0303  1 !++
;  241     0304  1 ! FUNCTIONAL DESCRIPTION:
;  242     0305  1 !
;  243     0306  1 ! Initialize read Formatted User data formatter (UDF)
;  244     0307  1 !
;  245     0308  1 ! CALLING SEQUENCE:
;  246     0309  1 !
;  247     0310  1 !       JSB FOR$$UDF_RF0
;  248     0311  1 !
;  249     0312  1 ! FORMAL PARAMETERS:
;  250     0313  1 !
;  251     0314  1 !       NONE
;  252     0315  1 !
;  253     0316  1 ! IMPLICIT INPUTS:
;  254     0317  1 !
;  255     0318  1 !       CCB                     Pointer to current logical unit block
;  256     0319  1 !
;  257     0320  1 !       ISB$B_STTM_TYPE         I/O statement type code - set by
;  258     0321  1 !                               each I/O statement initialization
;  259     0322  1 !
;  260     0323  1 ! IMPLICIT OUTPUTS:
;  261     0324  1 !
;  262     0325  1 !       LUB$A_BUF_BEG           Adr. of first byte of input data buffer
;  263     0326  1 !       LUB$A_BUF_PTR           Adr. of next byte of input
;  264     0327  1 !                               data buffer
;  265     0328  1 !       LUB$A_BUF_HIGH          Adr. of high water byte in input buffer on this
;  266     0329  1 !                               I/O statement
;  267     0330  1 !       LUB$A_BUF_END           Adr. +1 of last char position allocated
;  268     0331  1 !                               to input buffer
;  269     0332  1 !
;  270     0333  1 ! ROUTINE VALUE:
;  271     0334  1 ! COMPLETION CODES:
;  272     0335  1 !
;  273     0336  1 !       NONE
;  274     0337  1 !
;  275     0338  1 ! SIDE EFFECTS:
;  276     0339  1 !
;  277     0340  1 !       Initializes array AA_IN_CVT upon first entry.
;  278     0341  1 !
;  279     0342  1 !--
;  280     0343  1
;  281     0344  2     BEGIN
;  282     0345  2
;  283     0346  2     EXTERNAL REGISTER
;  284     0347  2         CCB : REF $FOR$CCB_DECL;
;  285     0348  2
;  286     0349  2     !+
;  287     0350  2     ! Initialize Record processing level of abstraction.
;  288     0351  2     ! Set pointer to current (LUB$A_BUF_PTR)  and last+1
;  289     0352  2     ! (LUB$A_BUF_END) character position for user data in
;  290     0353  2     ! input buffer
;  291     0354  2     !-
;  292     0355  2
;  293     0356  2     JSB_RECO (FOR$$AA_REC_PRO + .FOR$$AA_REC_PRO [.CCB [ISB$B_STTM_TYPE] - ISB$K_FORSTTYLO + 1]);
```

```
;  294     0357  2
;  295     0358  2      !+
;  296     0359  2      ! Initialize character pointer to first position for user
;  297     0360  2      ! data in input buffer - needed only for T AND $ formats
;  298     0361  2      !-
;  299     0362  2
;  300     0363  2      CCB [LUB$A_BUF_BEG] = .CCB [LUB$A_BUF_PTR];
;  301     0364  2
;  302     0365  2      !+
;  303     0366  2      ! Initialize Format interpreter
;  304     0367  2      !-
;  305     0368  2
;  306     0369  2      FOR$$FMT_INTRPO ();
;  307     0370  2
;  308     0371  2      !+
;  309     0372  2      ! Initialize character pointer to highest position written in
;  310     0373  2      ! user data buffer for this record.  T format may position to
;  311     0374  2      ! the left.
;  312     0375  2      !-
;  313     0376  2
;  314     0377  2      CCB [LUB$A_BUF_HIGH] = .CCB [LUB$A_BUF_PTR];
;  315     0378  2
;  316     0379  2      !+
;  317     0380  2      ! All other ISB locations and flags have already been
;  318     0381  2      ! initialized to 0 or a specified value by the I/O statement
;  319     0382  2      ! initialization for this I/O statement.
;  320     0383  2      !-
;  321     0384  2
;  322     0385  2      !+
;  323     0386  2      ! If array of conversion routine addresses has been intialized, then
;  324     0387  2      ! return.  Otherwise, initialize it.
;  325     0388  2      !-
;  326     0389  2
;  327     0390  2      IF .CVT_INIT
;  328     0391  2      THEN
;  329     0392  2          RETURN;
;  330     0393  2
;  331     0394  2      !+
;  332     0395  2      ! Store the conversion routine addresses in AA_IN_CVT.
;  333     0396  2      !-
;  334     0397  2
;  335     0398  2      AA_IN_CVT [_L - _L] = OTS$CVT_TL_L;          ! L format integer conversion
;  336     0399  2      AA_IN_CVT [_O - _L] = OTS$CVT_TO_L;          ! O format integer conversion
;  337     0400  2      AA_IN_CVT [_I - _L] = OTS$CVT_TI_L;          ! I format integer conversion
;  338     0401  2      AA_IN_CVT [_Z - _L] = OTS$CVT_TZ_L;          ! Z format integer conversion
;  339     0402  2      AA_IN_CVT [DSC$K_DTYPE_F] = OTS$CVT_T_F;     ! F_floating conversion
;  340     0403  2      AA_IN_CVT [DSC$K_DTYPE_D] = OTS$CVT_T_D;     ! D_floating conversion
;  341     0404  2      AA_IN_CVT [DSC$K_DTYPE_G] = OTS$CVT_T_G;     ! G_floating conversion
;  342     0405  2      AA_IN_CVT [DSC$K_DTYPE_H] = OTS$CVT_T_H;     ! H_floating conversion
;  343     0406  2      CVT_INIT = 1;                                ! Set initialized flag
;  344     0407  2
;  345     0408  2      RETURN;
;  346     0409  1      END;                                         ! End of FOR$$UDF_RF0 routine

                                                                    .TITLE  FOR$$UDF_RF FORTRAN Read Formatted UDF
                                                                    .IDENT  \1-043\
```

```
                                                        .PSECT  _FOR$DATA,NOEXE,  PIC,2

                                                00000 AA_IN_CVT:
                                                        .BLKB   116
                                      00000000  00074 CVT_INIT:
                                                        .LONG   0                                                       ;

                                                        .PSECT  _FOR$CODE,NOWRT,  SHR,  PIC,2

04  00  00  00  00  00  00  80  02  01  81  00  00  00  80  00000 P.AAA:  .BYTE   -128, 0, 0, 0, -127, 1, 2, -128, 0, 0, 0, - ;
00  E8  E8  E8  E8  E8  E8  E8  C7  86  00  00  00  00  45  0000F                0, 0, 0, 4, 69, 0, 0, 0, 0, -122, -57, -    ;
80  80  80  80  00  00  00  00  00  E9  E9  E9  E9  E9  E9  0001E                -24, -24, -24, -24, -24, -24, -24, 0, -     ;
                        80  80  80  80  00  00  00  00  80  0002D                -23, -23, -23, -23, -23, -23, 0, 0, 0, 0, - ;
                                                                                0, -128, -128, -128, -128, -128, 0, 0, 0, - ;
                                                                                0, -128, -128, -128, -128                   ;

                                                RF_ACT=                 P.AAA
                                                        .EXTRN  FOR$$AA_REC_PRO
                                                        .EXTRN  FOR$$AA_REC_PR1
                                                        .EXTRN  OTS$CVT_T_F, OTS$CVT_T_D
                                                        .EXTRN  OTS$CVT_T_G, OTS$CVT_T_H
                                                        .EXTRN  OTS$CVT_TL_L, OTS$CVT_TO_L
                                                        .EXTRN  OTS$CVT_TI_L, OTS$CVT_TZ_L
                                                        .EXTRN  FOR$$FMT_INTRP0
                                                        .EXTRN  FOR$$FMT_INTRP1
                                                        .EXTRN  FOR$$SIGNAL, FOR$$SIGNAL_STO

                                                        .EXTRN  FOR$$SIGNAL, FOR$$SIGNAL_STO

                        50      FF71    CB  9A 00000 FOR$$UDF_RF0::
                                                        MOVZBL  -143(CCB), R0                                           ; 0356
                        50 00000000G0040    D0 00005    MOVL    FOR$$AA_REC_PRO[R0], R0                                 ;
                           00000000G0040    16 0000D    JSB     FOR$$AA_REC_PRO[R0]                                     ;
                    BC  AB          B0  AB  D0 00014    MOVL    -80(CCB), -68(CCB)                                      ; 0363
                           00000000G    00  16 00019    JSB     FOR$$FMT_INTRP0                                         ; 0369
                    C0  AB          B0  AB  D0 0001F    MOVL    -80(CCB), -64(CCB)                                      ; 0377
                        5F 00000000'    EF  E8 00024    BLBS    CVT_INIT, 1$                                            ; 0390
          00000000'     EF 00000000G    00  9E 0002B    MOVAB   OTS$CVT_TL_L, AA_IN_CVT                                 ; 0398
          00000000'     EF 0000C000G    00  9E 00036    MOVAB   OTS$CVT_TO_L, AA_IN_CVT+4                               ; 0399
          00000000'     EF 00000000G    00  9E 00041    MOVAB   OTS$CVT_TI_L, AA_IN_CVT+8                               ; 0400
          00000000'     EF 00000000G    00  9E 0004C    MOVAB   OTS$CVT_TZ_L, AA_IN_CVT+12                              ; 0401
          00000000'     EF 00000000G    00  9E 00057    MOVAB   OTS$CVT_T_F, AA_IN_CVT+40                               ; 0402
          00000000'     EF 00000000G    00  9E 00062    MOVAB   OTS$CVT_T_D, AA_IN_CVT+44                               ; 0403
          00000000'     EF 00000000G    00  9E 0006D    MOVAB   OTS$CVT_T_G, AA_IN_CVT+108                              ; 0404
          00000000'     EF 00000000G    00  9E 00078    MOVAB   OTS$CVT_T_H, AA_IN_CVT+112                              ; 0405
          00000000'     EF              01  D0 00083    MOVL    #1, CVT_INIT                                            ; 0406
                                            05 0008A 1$:  RSB                                                           ; 0409

; Routine Size:  139 bytes,    Routine Base:  _FOR$CODE + 0036

;  347          0410  1
```

```
;  349     0411  1 GLOBAL ROUTINE FOR$$UDF_RF1 (                        ! Format one user input element
;  350     0412  1     ELEM_TYPE,                                      ! Type code of user I/O list element
;  351     0413  1     ELEM_SIZE,                                      ! No. of addressable units in element
;  352     0414  1     ELEM_ADR)                                       ! Adr. of element
;  353     0415  1     : CALL_CCB NOVALUE =
;  354     0416  1
;  355     0417  1 !++
;  356     0418  1 ! FUNCTIONAL DESCRIPTION:
;  357     0419  1 !
;  358     0420  1 !       FOR$$UDF_RF1  extracts the next field (W characters fromkt
;  359     0421  1 !       format statement, or up to next comma in input buffer, or end of
;  360     0422  1 !       input buffer, whichever occurs first) from the input buffer and
;  361     0423  1 !       converts it according to the type specified by the format
;  362     0424  1 !       statement and the size specified by the data type of the user
;  363     0425  1 !       I/O list element.
;  364     0426  1 !       FOR$UDF_RF1 and the format interpreter
;  365     0427  1 !       (FOR$$FMT_INTRP1) interpret all format codes until the
;  366     0428  1 !       first I/O list element transmitting format code is
;  367     0429  1 !       encountered and then continues up to but not including the next
;  368     0430  1 !       data transmitting format code.
;  369     0431  1 !
;  370     0432  1 !       FOR$$UDF_RF1 is also called by FOR$$UDF_RF9 if and only if
;  371     0433  1 !       there were no I/O list items to transmit, thereby causing the
;  372     0434  1 !       non-data transmitting format codes to be executed.
;  373     0435  1 !
;  374     0436  1 ! CALLING SEQUENCE:
;  375     0437  1 !
;  376     0438  1 !       CALL FOR$$UDF_RF1 (elem_type.rlu.v, elem_size.rlu.v, elem_adr.wx.r)
;  377     0439  1 !
;  378     0440  1 ! FORMAL PARAMETERS:
;  379     0441  1 !
;  380     0442  1 !       ELEM_TYPE.rlu.v         Type code of user I/O list
;  381     0443  1 !                               element. Form: ELEM_TYPE_x
;  382     0444  1 !                               x = B,W,L,WU,LU,F,D,G,H,FC,DC,GC or T.
;  383     0445  1 !                               If zero, then this is an end-of-list
;  384     0446  1 !                               call from FOR$$UDF_RF9.
;  385     0447  1 !       ELEM_SIZE.rlu.v         Size of user I/O list element
;  386     0448  1 !                               in addressable machine units (VAX, bytes)
;  387     0449  1 !       ELEM_ADR.wx.r           Adr. of user I/O list element
;  388     0450  1 !                               x = datatype
;  389     0451  1 !
;  390     0452  1 !
;  391     0453  1 ! IMPLICIT INPUTS:
;  392     0454  1 !
;  393     0455  1 !       CCB                     Pointer to current logical unit block
;  394     0456  1 !       ISB$B_STTM_TYPE         I/O statement type code - set by each
;  395     0457  1 !                               I/O statement initialization
;  396     0458  1 !
;  397     0459  1 !        The following ISB locations are set only by previous calls to
;  398     0460  1 !        FOR$$UDF_RF{0,1}, i.e., are effectively OWN.
;  399     0461  1 !
;  400     0462  1 !       LUB$A_BUF_BEG           Pointer to first char. position in
;  401     0463  1 !                               user data part of input buffer
;  402     0464  1 !       LUB$A_BUF_PTR           Pointer to next char. position
;  403     0465  1 !                               in user data part of input buffer
;  404     0466  1 !       LUB$A_BUF_END           Pointer to last+1 char. position
;  405     0467  1 !                               in user data part of input buffer
```

```
;  406          0468  1 !       The following ISB locations are set by the format interpreter
;  407          0469  1 !        (FOR$$FMT_INTRP1) which this module calls:
;  408          0470  1 !
;  409          0471  1 !       ISB$A_FMT_PTR           Pointer to next char. position
;  410          0472  1 !                               in user data part of input buffer
;  411          0473  1 !                               Used only in H format.
;  412          0474  1 !       ISB$W_FMT_W             Field width (w)
;  413          0475  1 !       ISB$B_FMT_D             No. of fraction digits (d)
;  414          0476  1 !       ISB$B_FMT_E             No. of exponent characters (e)
;  415          0477  1 !       ISB$B_FMT_P             Signed scale factor (p)
;  416          0478  1 !
;  417          0479  1 ! IMPLICIT OUTPUTS:
;  418          0480  1 !
;  419          0481  1 !       ISB$A_FMT_PTR           Pointer to next char. position
;  420          0482  1 !                               in compiled format character string
;  421          0483  1 !                               Changed only for H format.
;  422          0484  1 !
;  423          0485  1 !        The following ISB locations are set only by previous calls
;  424          0486  1 !        to FOR$$UDF_RF(0,1), i.e., are effectively OWN.
;  425          0487  1 !
;  426          0488  1 !       LUB$A_BUF_PTR           Pointer to next char. position
;  427          0489  1 !                               in user data part of input buffer
;  428          0490  1 !       ISB$B_ERR_NO            FOR$_INPCONERR (43='INPUT CONVERSION ERROR') -
;  429          0491  1 !                               overflowed field is filled with *'s.
;  430          0492  1 !                               FOR$_FORVARMIS (61='FORMAT/VARIABLE-TYPE MISMATCH')
;  431          0493  1 !
;  432          0494  1 ! FUNCTIONAL VALUE:
;  433          0495  1 !
;  434          0496  1 !       NONE
;  435          0497  1 !
;  436          0498  1 ! SIDE EFFECTS:
;  437          0499  1 !
;  438          0500  1 !--
;  439          0501  1
;  440          0502  2     BEGIN
;  441          0503  2
;  442          0504  2     EXTERNAL REGISTER
;  443          0505  2         CCB : REF $FOR$CCB_DECL;
;  444          0506  2
;  445          0507  2     MAP
;  446          0508  2         ELEM_ADR : REF VECTOR;                  ! element is call-by-reference
;  447          0509  2
;  448          0510  2     GLOBAL REGISTER
;  449          0511  2         EL_SIZE = 10,                           ! Element size
;  450          0512  2         DT_SEEN = 9,                            ! Data transmitter seen
;  451          0513  2         FMT_CODE = 8 : BLOCK [1, LONG];         ! Format code
;  452          0514  2
;  453          0515  2     LOCAL
;  454          0516  2         ACT : BLOCK [1, LONG],                  ! Action table entry for format code
;  455          0517  2         BUFPTR,                                 ! Input buffer pointer from ISB
;  456          0518  2         FMT_W,                                  ! Input field width from ISB
;  457          0519  2         DSC : BLOCK [8, BYTE];                  ! Static string desciptor for
;  458          0520  2                                                 ! output field
;  459          0521  2
;  460          0522  2
;  461          0523  2     EL_SIZE = .ELEM_SIZE;                       ! Fetch first argument
;  462          0524  2
```

```
;  463     0525  2      !+
;  464     0526  2      ! Set DT_SEEN to zero unless this is a call from FOR$$UDF_RF9
;  465     0527  2      ! (no items in I/O list) in which case set DT_SEEN to 1 so that
;  466     0528  2      ! we stop on the next data transmitter.
;  467     0529  2      !-
;  468     0530  2
;  469     0531  2      IF .ELEM_TYPE EQL 0 THEN DT_SEEN = 1 ELSE DT_SEEN = 0;
;  470     0532  2
;  471     0533  2      !+
;  472     0534  2      ! Execute format items until we come across one which calls for
;  473     0535  2      ! an I/O list item that we don't have.
;  474     0536  2      !-
;  475     0537  2
;  476     0538  2      WHILE 1 DO
;  477     0539  2
;  478     0540  2      !+
;  479     0541  2      ! Get next format code requiring input interpretation:
;  480     0542  2      !   1.  If we are in a repeated format code (nI, not n(I)),
;  481     0543  2      !       save a call to the format interpreter by getting the
;  482     0544  2      !       stored code ourselves.  If this would mean that we
;  483     0545  2      !       exit, do so without decrementing the repeat count.
;  484     0546  2      !
;  485     0547  2      !   2.  Otherwise, call the format interpreter to get the next
;  486     0548  2      !       format code.
;  487     0549  2      !
;  488     0550  2      !   3.  If this format code is a data transmitter (or : or EOF),
;  489     0551  2      !       and we have already seen a data transmitter, exit.  It
;  490     0552  2      !       will still be there if we come back.
;  491     0553  2      !
;  492     0554  2      ! Dispatch on format code and select appropriate actions.
;  493     0555  2      !-
;  494     0556  2
;  495     0557  3          BEGIN
;  496     0558  3
;  497     0559  3          IF .CCB [ISB$W_FMT_REP] GTR 1 AND .CCB [ISB$B_FMT_CODE] LSSU _DA
;  498     0560  3          THEN
;  499     0561  4              BEGIN
;  500     0562  4              FMT_CODE = .CCB [ISB$B_FMT_CODE];
;  501     0563  4              ACT = .RF_ACT [.FMT_CODE];
;  502     0564  4
;  503     0565  4              IF .DT_SEEN
;  504     0566  4              THEN
;  505     0567  4
;  506     0568  4                  IF .ACT [RF_EOLST] THEN EXITLOOP;
;  507     0569  4
;  508     0570  4              CCB [ISB$W_FMT_REP] = .CCB [ISB$W_FMT_REP] - 1;
;  509     0571  4              END
;  510     0572  3          ELSE
;  511     0573  4              BEGIN
;  512     0574  4
;  513     0575  4              !+
;  514     0576  4              ! If DT_SEEN is true, then we only want to know if the next
;  515     0577  4              ! format code would transmit a data item.  Rather than have
;  516     0578  4              ! the high overhead of calling the format interpreter, we
;  517     0579  4              ! can look ahead into the format for this information.  We
;  518     0580  4              ! can't make a 100% determination, so if the format is not
;  519     0581  4              ! an "EOLST" type, call the format interpreter anyway.
```

```
;  520     0582  4              ! This is a speed optimization.  If necessary, the code
;  521     0583  4              ! between the "!**"s can be removed with no functionality loss.
;  522     0584  4              !-
;  523     0585  4
;  524     0586  4              !**
;  525     0587  4
;  526     0588  4              IF .DT_SEEN
;  527     0589  4              THEN
;  528     0590  5                  BEGIN
;  529     0591  5
;  530     0592  5                  LOCAL
;  531     0593  5                      P;                              ! Pointer into format
;  532     0594  5
;  533     0595  5                  P = .CCB [ISB$A_FMT_PTR];
;  534     0596  5                  FMT_CODE = CH$RCHAR (.P);           ! Get next format code
;  535     0597  5                  FMT_CODE [V_FMT_REPRE] = 0;         ! Clear bit for comparison
;  536     0598  5                  ACT = .RF_ACT [.FMT_CODE];
;  537     0599  5
;  538     0600  5                  IF .ACT [RF_EOLST] THEN EXITLOOP;       ! End of list type
;  539     0601  5
;  540     0602  4                  END;
;  541     0603  4
;  542     0604  4              !**
;  543     0605  4              FOR$$FMT_INTRP1 ();                     ! Call format interpreter.
;  544     0606  4                                                      ! Implicit arguments are EL_SIZE
;  545     0607  4                                                      ! and DT_SEEN.  Implicit result
;  546     0608  4                                                      ! is FMT_CODE.
;  547     0609  4              ACT = .RF_ACT [.FMT_CODE];
;  548     0610  4
;  549     0611  4              IF .DT_SEEN AND .ACT [RF_EOLST] THEN EXITLOOP;
;  550     0612  4
;  551     0613  3                  END;
;  552     0614  3
;  553     0615  3              !+
;  554     0616  3              ! All data generating format codes (A,L,O,Z,I
;  555     0617  3              ! F,E,G,D, except Q plus H):
;  556     0618  3              ! Setup string descriptor to field of width W.
;  557     0619  3              ! (ISB$W_FMT_W) and next char position
;  558     0620  3              ! for output (LUB$A_BUF_PTR) in
;  559     0621  3              ! output buffer. Check for field extending beyond
;  560     0622  3              ! end of buffer and set DSC[DSC$W_LENGTH] in
;  561     0623  3              ! string descriptor to no. of characters which remain
;  562     0624  3              ! in input buffer if would run off the end.
;  563     0625  3              !-
;  564     0626  3
;  565     0627  3              IF .ACT [RF_CHECKW]
;  566     0628  3              THEN
;  567     0629  4                  BEGIN
;  568     0630  4                  DSC [DSC$W_LENGTH] = .CCB [ISB$W_FMT_W];
;  569     0631  4                  DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  570     0632  4                  DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  571     0633  4                  DSC [DSC$A_POINTER] = .CCB [LUB$A_BUF_PTR];
;  572     0634  4                  CCB [LUB$A_BUF_PTR] = CH$PLUS (.CCB [LUB$A_BUF_PTR], .CCB [ISB$W_FMT_W]);
;  573     0635  4
;  574     0636  5                  IF (.CCB [LUB$A_BUF_PTR] GTR .CCB [LUB$A_BUF_END])
;  575     0637  4                  THEN
;  576     0638  5                      BEGIN                           ! Field would extend beyond end of buffer - reset
```

```
;  577     0639  5                DSC [DSC$W_LENGTH] = MAX (CH$DIFF (.CCB [LUB$A_BUF_END], .DSC [DSC$A_POINTER]), 0);
;  578     0640  4                END;
;  579     0641  4
;  580     0642  4            !+
;  581     0643  4            ! Short input field check, i.e., a field terminated
;  582     0644  4            ! by an explicit comma in the data earlier
;  583     0645  4            ! than the width of field specified by the format statement.
;  584     0646  4            ! If a short field, reduce to include up to but not including
;  585     0647  4            ! the comma, but advance character pointer (LUB$A_BUF_PTR)
;  586     0648  4            ! beyond the comma, so it will not be found on next element.
;  587     0649  4            ! A zero length field is treated as a string of spaces.
;  588     0650  4            !-
;  589     0651  4
;  590     0652  4            IF .ACT [RF_SHORT]
;  591     0653  4            THEN
;  592     0654  5                BEGIN
;  593     0655  5
;  594     0656  5                LOCAL
;  595     0657  5                    P;                              ! temporary character pointer
;  596     0658  5
;  597     0659  5                P = CH$FIND_CH (.DSC [DSC$W_LENGTH], .DSC [DSC$A_POINTER], %C',');
;  598     0660  5
;  599     0661  5                IF .P NEQ 0
;  600     0662  5                THEN
;  601     0663  6                    BEGIN
;  602     0664  6                    DSC [DSC$W_LENGTH] = CH$DIFF (.P, .DSC [DSC$A_POINTER]);
;  603     0665  6                    CCB [LUB$A_BUF_PTR] = CH$PLUS (.P, 1);
;  604     0666  5                    END;
;  605     0667  5
;  606     0668  4                END;                                ! End of short field check
;  607     0669  4
;  608     0670  3            END;                                    ! End of CHECKW
;  609     0671  3
;  610     0672  3            CASE .ACT [RF_DISPAT] FROM 0 TO 9 OF
;  611     0673  3                SET
;  612     0674  3
;  613     0675  3                [0] :
;  614     0676  3
;  615     0677  3                    !+
;  616     0678  3                    ! Colon:  Only get here if not end of user I/O list,
;  617     0679  3                    ! so keep on looking for a data transmitting format code.
;  618     0680  3                    !-
;  619     0681  3
;  620     0682  3                    ;                               ! do nothing
;  621     0683  3
;  622     0684  3                [1] :
;  623     0685  3
;  624     0686  3                    !+
;  625     0687  3                    ! End of format or / format code seen:
;  626     0688  3                    ! Call record processing level (REC_PR1) for appropriate
;  627     0689  3                    ! statement type.  \\ Note that we now allow direct access
;  628     0690  3                    ! files to read more than one record.  \\
;  629     0691  3                    ! Initialize all input buffer pointer for next record
;  630     0692  3                    ! in this I/O statement, e.g., ISB$A_BUF_(BEG,PTR,END)
;  631     0693  3                    ! and ISB$V_DOLLAR = 0.
;  632     0694  3                    !-
;  633     0695  3
```

```
;  634          0696  3                DO_READ (FOR$$AA_REC_PR1 + .FOR$$AA_REC_PR1 [.CCB [ISB$B_STTM_TYPE] - ISB$K_FORSTTYLO + 1]);
;  635          0697  3
;  636          0698  3            [2] :
;  637          0699  3
;  638          0700  3                !+
;  639          0701  3                ! Dollar sign: Do nothing for read.  $ only affects write
;  640          0702  3                !-
;  641          0703  3
;  642          0704  3            ;                                       ! do nothing
;  643          0705  3
;  644          0706  3            [3] :
;  645          0707  3
;  646          0708  3                !+
;  647          0709  3                ! No longer used.
;  648          0710  3                !-
;  649          0711  3
;  650          0712  3            ;
;  651          0713  3
;  652          0714  3            [4] :
;  653          0715  3
;  654          0716  3                !+
;  655          0717  3                ! nX
;  656          0718  3                ! Move right n characters.  This format code is no longer
;  657          0719  3                ! generated, but it must continue to work for old programs.
;  658          0720  3                !-
;  659          0721  3
;  660          0722  3                CCB [LUB$A_BUF_PTR] = CH$PLUS (.CCB [LUB$A_BUF_PTR], .CCB [ISB$W_FMT_W]);
;  661          0723  3
;  662          0724  3            [5] :
;  663          0725  3
;  664          0726  3                !+
;  665          0727  3                ! nHccccc: Holerith - copy n (DSC$W_LENGTH) chars
;  666          0728  3                ! from input buffer to format array. Update format
;  667          0729  3                ! character pointer (ISB$A_FMT_PTR).  Format array is
;  668          0730  3                ! blank padded if data in array is shorter than format.
;  669          0731  3                !-
;  670          0732  3
;  671          0733  3                CCB [ISB$A_FMT_PTR] = COPY_CHAR (.DSC [DSC$W_LENGTH], .DSC [DSC$A_POINTER],
;  672          0734  3                    .CCB [ISB$W_FMT_W], .CCB [ISB$A_FMT_PTR]);
;  673          0735  3
;  674          0736  3            [6] :
;  675          0737  3
;  676          0738  3                !+
;  677          0739  3                ! Q format -  return no. of character positions remaining
;  678          0740  3                ! in input buffer (ie., in record) as an integer.
;  679          0741  3                ! Size of integer depends on size of user I/O list element data type.
;  680          0742  3                ! If user element type is not integer, SIGNAL and store
;  681          0743  3                ! into low order 32 bits.
;  682          0744  3                ! Then exit loop and return to user program
;  683          0745  3                !-
;  684          0746  3
;  685          0747  4                BEGIN
;  686          0748  4
;  687          0749  4                IF .ELEM_TYPE LSSU DSC$K_DTYPE_BU OR .ELEM_TYPE GTRU DSC$K_DTYPE_Q
;  688          0750  4                THEN
;  689          0751  4                    CCB [ISB$B_ERR_NO] = FOR$K_FORVARMIS;
;  690          0752  4
```

```
;  691          0753  4                    (.ELEM_ADR)<0, MINU (4, .EL_SIZE)*%BPUNIT, 0> = MAX (0,
;  692          0754  4                        CH$DIFF (.CCB [LUB$A_BUF_END],
;  693          0755  4                                 .CCB [LUB$A_BUF_PTR]));
;  694          0756  4                    DT_SEEN = 1;
;  695          0757  3                    END;                                ! End of Q input
;  696          0758  3
;  697          0759  3                [7] :
;  698          0760  3
;  699          0761  3                    !+
;  700          0762  3                    ! nAw.d and nA formats: Copy string from input field to user data element.
;  701          0763  3                    ! Copy right-most characters up to datatype size and
;  702          0764  3                    ! blank fill remainder if any.
;  703          0765  3                    !-
;  704          0766  3
;  705          0767  4                    BEGIN
;  706          0768  4
;  707          0769  4                    !+
;  708          0770  4                    ! If the element is greater than the format width,
;  709          0771  4                    ! then move the characters and blank fill.
;  710          0772  4                    !-
;  711          0773  4
;  712          0774  4                    IF .EL_SIZE GTRU .DSC [DSC$W_LENGTH]
;  713          0775  4                    THEN
;  714          0776  4                        COPY_CHAR (.DSC [DSC$W_LENGTH],
;  715          0777  4                            .DSC [DSC$A_POINTER], .EL_SIZE, .ELEM_ADR)
;  716          0778  4                    ELSE
;  717          0779  5                        BEGIN
;  718          0780  5
;  719          0781  5                        !+
;  720          0782  5                        ! Element size is less than or equal to format width.
;  721          0783  5                        ! If less than, move rightmost characters only.  Use
;  722          0784  5                        ! non-character moves if possible.
;  723          0785  5                        !-
;  724          0786  5
;  725          0787  5                        LOCAL
;  726          0788  5                            ELEM_PTR,
;  727          0789  5                            BUF_PTR;
;  728          0790  5
;  729          0791  5                        IF .EL_SIZE LSSU .DSC [DSC$W_LENGTH]
;  730          0792  5                        THEN
;  731          0793  6                            BUF_PTR = .DSC [DSC$A_POINTER] + (.DSC [DSC$W_LENGTH] - .EL_SIZE)
;  732          0794  5                        ELSE
;  733          0795  5                            BUF_PTR = .DSC [DSC$A_POINTER];
;  734          0796  5
;  735          0797  5                        ELEM_PTR = .ELEM_ADR;
;  736          0798  5
;  737          0799  5                        CASE .EL_SIZE FROM 0 TO 8 OF
;  738          0800  5                            SET
;  739          0801  5
;  740          0802  5                            [8] :
;  741          0803  6                                BEGIN
;  742          0804  6                                COPY_QUAD_A (BUF_PTR, ELEM_PTR);
;  743          0805  5                                END;
;  744          0806  5
;  745          0807  5                            [7] :
;  746          0808  6                                BEGIN
;  747          0809  6                                COPY_LONG_A (BUF_PTR, ELEM_PTR);
```

```
;  748          0810  6                    COPY_WORD_A (BUF_PTR, ELEM_PTR);
;  749          0811  6                    COPY_BYTE_A (BUF_PTR, ELEM_PTR);
;  750          0812  5                    END;
;  751          0813  5
;  752          0814  5                [6] :
;  753          0815  6                    BEGIN
;  754          0816  6                    COPY_LONG_A (BUF_PTR, ELEM_PTR);
;  755          0817  6                    COPY_WORD_A (BUF_PTR, ELEM_PTR);
;  756          0818  5                    END;
;  757          0819  5
;  758          0820  5                [5] :
;  759          0821  6                    BEGIN
;  760          0822  6                    COPY_LONG_A (BUF_PTR, ELEM_PTR);
;  761          0823  6                    COPY_BYTE_A (BUF_PTR, ELEM_PTR);
;  762          0824  5                    END;
;  763          0825  5
;  764          0826  5                [4] :
;  765          0827  6                    BEGIN
;  766          0828  6                    COPY_LONG_A (BUF_PTR, ELEM_PTR);
;  767          0829  5                    END;
;  768          0830  5
;  769          0831  5                [3] :
;  770          0832  6                    BEGIN
;  771          0833  6                    COPY_WORD_A (BUF_PTR, ELEM_PTR);
;  772          0834  6                    COPY_BYTE_A (BUF_PTR, ELEM_PTR);
;  773          0835  5                    END;
;  774          0836  5
;  775          0837  5                [2] :
;  776          0838  6                    BEGIN
;  777          0839  6                    COPY_WORD_A (BUF_PTR, ELEM_PTR);
;  778          0840  5                    END;
;  779          0841  5
;  780          0842  5                [1] :
;  781          0843  6                    BEGIN
;  782          0844  6                    COPY_BYTE_A (BUF_PTR, ELEM_PTR);
;  783          0845  5                    END;
;  784          0846  5
;  785          0847  5                [0] :
;  786          0848  5                    ;
;  787          0849  5
;  788          0850  5                [OUTRANGE] :
;  789          0851  5                    MOVE_CHAR (.EL_SIZE, .BUF_PTR, .ELEM_PTR);
;  790          0852  5                TES;
;  791          0853  5
;  792          0854  4                END;
;  793          0855  4
;  794          0856  4            DT_SEEN = 1;
;  795          0857  3            END;
;  796          0858  3
;  797          0859  3        [8] :
;  798          0860  3
;  799          0861  3            !+
;  800          0862  3            ! All integer formats (L,O,I,Z) output:
;  801          0863  3            ! 1) Check data type. If user I/O list element is not integer (B,W,L,WU,LU),
;  802          0864  3            ! SIGNAL FOR$_FORVARMIS (61='FORMAT VARIABLE-TYPE MISMATCH').
;  803          0865  3            ! unless format is not I; else store one longword.
;  804          0866  3            !-
```

```
;  805     0867  3
;  806     0868  4                    BEGIN
;  807     0869  4
;  808     0870  4                    LOCAL
;  809     0871  4                        S;                              ! No. of addressable units in
;  810     0872  4
;  811     0873  4                    ! user I/O list element.
;  812     0874  4
;  813     0875  4                    !+
;  814     0876  4                    ! Compensate if extended format Iw.m, etc., which makes
;  815     0877  4                    ! no difference here.
;  816     0878  4                    !-
;  817     0879  4
;  818     0880  4                    IF .FMT_CODE GEQU XO THEN FMT_CODE = .FMT_CODE - (_L + 3) ELSE FMT_CODE = .FMT_CODE - _L;
;  819     0881  4
;  820     0882  4                    !-
;  821     0883  4
;  822     0884  5                    IF (.ELEM_TYPE GEQU DSC$K_DTYPE_Q) AND (.FMT_CODE EQLU (_L - _L) OR .FMT_CODE EQLU (_I - _L)
;  823     0885  4                    THEN
;  824     0886  5                        BEGIN
;  825     0887  5                        CCB [ISB$B_ERR_NO] = FOR$K_FORVARMIS;
;  826     0888  5                        S = %UPVAL;
;  827     0889  5                        END
;  828     0890  4                    ELSE
;  829     0891  4                        S = .EL_SIZE;
;  830     0892  4
;  831     0893  4                    !+
;  832     0894  4                    ! 2) Call appropriate library conversion routine
;  833     0895  4                    ! Sign extend (I,L) or zero-extend (O,Z) result (V).
;  834     0896  4                    ! If value could not fit, SIGNAL FOR$_INPCONERR
;  835     0897  4                    ! (64='INPUT CONVERSION ERROR' - low order bits stored correctly.
;  836     0898  4                    !-
;  837     0899  4
;  838     0900  4                    IF NOT (.AA_IN_CVT [.FMT_CODE]) (DSC, .ELEM_ADR, .S, .CCB [ISB$B_INP_FLAGS])
;  839     0901  4                    THEN
;  840     0902  4                        !+
;  841     0903  4                        ! If this is an indexed or internal file, then don't
;  842     0904  4                        ! try to put out a record number.
;  843     0905  4                        !-
;  844     0906  4
;  845     0907  5                        IF (.CCB [LUB$B_ORGAN] EQL LUB$K_ORG_INDEX) OR (.CCB [LUB$W_LUN] EQL LUB$K_LUN_ENCD)
;  846     0908  4                        THEN
;  847     0909  4                            FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEX, 1, DSC)
;  848     0910  4                        ELSE
;  849     0911  4                            FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEXREC, 2, DSC, .CCB [LUB$L_LOG_RECNO] - 1);
;  850     0912  4
;  851     0913  4                    DT_SEEN = 1;
;  852     0914  3                    END;                                ! End of L,O,I,Z input
;  853     0915  3
;  854     0916  3                [9] :
;  855     0917  3
;  856     0918  3                    !+
;  857     0919  3                    ! All Floating formats (F,E,G,D) input:
;  858     0920  3                    !-
;  859     0921  3
;  860     0922  4                    BEGIN
;  861     0923  4
```

```
;  862     0924  4                !+
;  863     0925  4                ! Call the appropriate conversion routine
;  864     0926  4                ! If the value did not fit in field, SIGNAL FOR$_INPCONERR
;  865     0927  4                ! (INPUT CONVERSION ERROR)
;  866     0928  4                !
;  867     0929  4                ! Store the floating value
;  868     0930  4                !-
;  869     0931  4
;  870     0932  4                !+
;  871     0933  4                ! Check for correct datatype
;  872     0934  4                !-
;  873     0935  4
;  874   P 0936  4                IF ONE_OF (.ELEM_TYPE, DSC$K_DTYPE_F, DSC$K_DTYPE_D,
;  875     0937  5                    DSC$K_DTYPE_G, DSC$K_DTYPE_H)
;  876     0938  4                THEN
;  877     0939  5                    BEGIN
;  878     0940  6                    IF NOT (.AA_IN_CVT [.ELEM_TYPE])
;  879     0941  5                        (DSC, .ELEM_ADR, .CCB [ISB$B_FMT_D], .CCB [ISB$B_FMT_P],
;  880     0942  5                            .CCB [ISB$B_INP_FLAGS])
;  881     0943  5                    THEN
;  882     0944  5                        !+
;  883     0945  5                        ! If this is an indexed or internal file, then don't
;  884     0946  5                        ! try to put out a record number.
;  885     0947  5                        !-
;  886     0948  5
;  887     0949  5                        IF (.CCB [LUB$B_ORGAN] EQL LUB$K_ORG_INDEX) OR
;  888     0950  6                            (.CCB [LUB$Q_LUN] EQL LUB$K_LUN_ENCD)
;  889     0951  5                        THEN
;  890     0952  5                            FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEX, 1, DSC)
;  891     0953  5                        ELSE
;  892     0954  5                            FOR$$SIGNAL (FOR$K_INPCONERR, FOR$_INVTEXREC, 2, DSC,
;  893     0955  5                                .CCB [LUB$L_LOG_RECNO] - 1);
;  894     0956  5                    END
;  895     0957  4                ELSE
;  896     0958  5                    BEGIN
;  897     0959  5                    !+
;  898     0960  5                    ! Datatype is not floating.  Convert as if F, store
;  899     0961  5                    ! correct size, and give "format/variable type mismatch"
;  900     0962  5                    ! error.
;  901     0963  5                    !-
;  902     0964  5
;  903     0965  5                    LOCAL
;  904     0966  5                        F_VALUE;
;  905     0967  5
;  906     0968  5                    OTS$CVT_T_F (DSC, F_VALUE, .CCB [ISB$B_FMT_D],
;  907     0969  5                        .CCB [ISB$B_FMT_P], .CCB [ISB$B_INP_FLAGS]);
;  908     0970  5                    (.ELEM_ADR)<0,MINU(4,.EL_SIZE)*%BPUNIT,0> = .F_VALUE;
;  909     0971  5                    CCB [ISB$B_ERR_NO] = FOR$K_FORVARMIS;
;  910     0972  4                    END;
;  911     0973  4
;  912     0974  4                !+
;  913     0975  4                ! Exit loop and return to user program
;  914     0976  4                !-
;  915     0977  4
;  916     0978  4                DT_SEEN = 1;
;  917     0979  3                END;                                ! End of F,E,G,D output
;  918     0980  3            TES;                                    ! End of CASE (entire loop)
```

```
;  919          0981  3
;  920          0982  2          END;                                    ! End of processing
;  921          0983  2
;  922          0984  2      RETURN;                                     ! Return from FOR$$UDF_RF1 routine
;  923          0985  1      END;                                        ! End of FOR$$UDF_RF1



                                    077C 00000          .ENTRY  FOR$$UDF_RF1, Save R2,R3,R4,R5,R6,R8,R9,R10 ; 0411
                                5E    0C   C2 00002          SUBL2   #12, SP
                                5A    08   AC D0 00005          MOVL    ELEM_SIZE, EL_SIZE                      ; 0523
                                54    04   AC D0 00009          MOVL    ELEM_TYPE, R4                           ; 0531
                                           03 12 0000D          BNEQ    1$
                                         0296 31 0000F          BRW     45$
                                           59 D4 00012 1$:      CLRL    DT_SEEN
                                01      8D   AB B1 00014 2$:      CMPW    -1T5(CCB), #1                           ; 0559
                                           1D 15 00018          BLEQ    4$
                                29      8F   AB 91 0001A          CMPB    -113(CCB), #41
                                           17 1E 0001E          BGEQU   4$
                                58      8F   AB 9A 00020          MOVZBL  -113(CCB), FMT_CODE                     ; 0562
                                55    FF16 CF48 9A 00024          MOVZBL  RF_ACT[FMT_CODE], ACT                   ; 0563
                                05         59 E9 0002A          BLBC    DT_SEEN, 3$                             ; 0565
                                           55 95 0002D          TSTB    ACT                                     ; 0568
                                           01 18 0002F          BGEQ    3$
                                              04 00031          RET
                                         8D   AB B7 00032 3$:      DECW    -115(CCB)                               ; 0570
                                           2D 11 00035          BRB     6$                                      ; 0559
                                16         59 E9 00037 4$:      BLBC    DT_SEEN, 5$                             ; 0588
                                50         80   AB D0 0003A          MOVL    -128(CCB), P                            ; 0595
                                58              60 9A 0003E          MOVZBL  (P), FMT_CODE                           ; 0596
                                58         80   8F 8A 00041          BICB2   #128, FMT_CODE                          ; 0597
                                55       FEF5 CF48 9A 00045          MOVZBL  RF_ACT[FMT_CODE], ACT                   ; 0598
                                           55 95 00048          TSTB    ACT                                     ; 0600
                                           01 18 0004D          BGEQ    5$
                                              04 0004F          RET
                                00000000G  00 16 00050 5$:      JSB     FOR$$FMT_INTRP1                         ; 0605
                                55       FEE4 CF48 9A 00056          MOVZBL  RF_ACT[FMT_CODE], ACT                   ; 0609
                                05         59 E9 0005C          BLBC    DT_SEEN, 6$                             ; 0611
                                           55 95 0005F          TSTB    ACT
                                           01 18 00061          BGEQ    6$
                                              04 00063          RET
                          4A               55    06 E1 00064 6$:      BBC     #6, ACT, 10$                            ; 0627
                                04    AE      89   AB B0 00068          MOVW    -119(CCB), DSC                          ; 0630
                                06    AE    010E   8F B0 0006D          MOVW    #270, DSC+2                             ; 0631
                                08    AE      B0   AB D0 00073          MOVL    -80(CCB), DSC+4                         ; 0633
                                50            89   AB 3C 00078          MOVZWL  -119(CCB), R0                           ; 0634
                                B0    AB            50 C0 0007C          ADDL2   R0, -80(CCB)
                                B4    AB            B0 AB D1 00080          CMPL    -80(CCB), -76(CCB)                      ; 0636
                                                    0E 15 00085          BLEQ    8$
                          50    B4    AB            08 AE C3 00087          SUBL3   DSC+4, -76(CCB), R0             ; 0639
                                                    02 18 0008D          BGEQ    7$
                                                    50 D4 0008F          CLRL    R0
                                04    AE                50 B0 00091 7$:      MOVW    R0, DSC
                                19            55        05 E1 00095 8$:      BBC     #5, ACT, 10$                    ; 0652
                          08    BE      04    AE        2C 3A 00099          LOCC    #44, DSC, @DSC+4                ; 0659
```

```
                                    02  12 0009F        BNEQ    9$                              ;
                                    51  D4 000A1        CLRL    R1                              ;
                                    51  D5 000A3 9$:    TSTL    P                               ; 0661
                                    0B  13 000A5        BEQL    10$                             ;
          04   AE        51     08  AE  A3 000A7        SUBW3   DSC+4, P, DSC                   ; 0664
                    B0   AB     01  A1  9E 000AD        MOVAB   1(R1), -80(CCB)                 ; 0665
   56            55      04         00  EF 000B2 10$:   EXTZV   #0, #4, ACT, R6                 ; 0672
                 09      00         56  CF 000B7        CASEL   R6, #0, #9                      ;
 FF59          FF59    0017       FF59     000BB 11$:   .WORD   2$-11$,-                        ;
 0087          0057    003D       0032     000C3                12$-11$,-                       ;
                       014E       010D     000CB                2$-11$,-                        ;
                                                                2$-11$,-                        ;
                                                                13$-11$,-                       ;
                                                                14$-11$,-                       ;
                                                                15$-11$,-                       ;
                                                                20$-11$,-                       ;
                                                                33$-11$,-                       ;
                                                                39$-11$                         ;
                                  FF42  31 000CF        BRW     2$                              ;
                         50   FF71  CB  9A 000D2 12$:   MOVZBL  -143(CCB), R0                   ; 0696
                         50 00000000G0040  D0 000D7     MOVL    FOR$$AA_REC_PR1[R0], R0         ;
                         50 00000000G0040  9E 000DF     MOVAB   FOR$$AA_REC_PR1[R0], R0         ;
                                 0000V  30 000E7        BSBW    DO_READ                         ;
                                  FF27  31 000EA        BRW     2$                              ;
                         50     89  AB  3C 000ED 13$:   MOVZWL  -119(CCB), R0                   ; 0722
                    B0   AB         50  C0 000F1        ADDL2   R0, -80(CCB)                    ;
                                  FF1C  31 000F5        BRW     2$                              ;
                                80  AB  DD 000F8 14$:   PUSHL   -128(CCB)                       ; 0734
                         7E     89  AB  3C 000FB        MOVZWL  -119(CCB), -(SP)                ;
                                10  AE  DD 000FF        PUSHL   DSC+4                           ; 0733
                         7E     10  AE  3C 00102        MOVZWL  DSC, -(SP)                      ;
                 0000V   CF         04  FB 00106        CALLS   #4, COPY_CHAR                   ;
                    80   AB         50  D0 0010B        MOVL    R0, -128(CCB)                   ;
                                  FF02  31 0010F        BRW     2$                              ;
                         02         54  D1 00112 15$:   CMPL    R4, #2                          ; 0749
                                    05  1F 00115        BLSSU   16$                             ;
                         09         54  D1 00117        CMPL    R4, #9                          ;
                                    05  1B 0011A        BLEQU   17$                             ;
                  FF70   CB         3D  90 0011C 16$:   MOVB    #61, -144(CCB)                  ; 0751
                         50         5A  D0 00121 17$:   MOVL    EL_SIZE, R0                     ; 0753
                         04         50  D1 00124        CMPL    R0, #4                          ;
                                    03  1B 00127        BLEQU   18$                             ;
                         50         04  D0 00129        MOVL    #4, R0                          ;
                         50         08  C4 0012C 18$:   MULL2   #8, R0                          ;
               51   B4   AB     B0  AB  C3 0012F        SUBL3   -80(CCB), -76(CCB), R1          ; 0755
                                    02  18 00135        BGEQ    19$                             ;
                                    51  D4 00137        CLRL    R1                              ;
 0C   BC         50      00         51  F0 00139 19$:   INSV    R1, #0, R0, @ELEM_ADR           ;
                                  0166  31 0013F        BRW     45$                             ; 0756
   5A     04     AE      10         00  ED 00142 20$:   CMPZV   #0, #16, DSC, EL_SIZE           ; 0774
                                    14  1E 00148        BGEQU   21$                             ;
                                0C  AC  DD 0014A        PUSHL   ELEM_ADR                        ; 0777
                                    5A  DD 0014D        PUSHL   EL_SIZE                         ;
                                10  AE  DD 0014F        PUSHL   DSC+4                           ;
                         7E     10  AE  3C 00152        MOVZWL  DSC, -(SP)                      ; 0776
                 0000V   CF         04  FB 00156        CALLS   #4, COPY_CHAR                   ;
                                  014A  31 0015B        BRW     45$                             ;
```

```
                5A      04   AE            10              00  ED 0015E 21$:    CMPZV   #0, #16, DSC, EL_SIZE                  ; 0791
                                           0E              00  1B 00164          BLEQU   22$                                    ;
                                           50        04    AE  3C 00166          MOVZWL  DSC, R0                                ; 0793
                                           50        5A    C2  0016A             SUBL2   EL_SIZE, R0                            ;
                                        52 50        08    AE  C1 0016D          ADDL3   DSC+4, R0, BUF_PTR                     ;
                                           52              04  11 00172          BRB     23$                                    ;
                                           52        08    AE  D0 00174 22$:     MOVL    DSC+4, BUF_PTR                         ; 0795
                                           53        0C    AC  D0 00178 23$:     MOVL    ELEM_ADR, ELEM_PTR                     ; 0797
                                     08    00              5A  CF 0017C          CASEL   EL_SIZE, #0, #8                        ; 0799
                0027          003C          0042        0128      00180 24$:     .WORD   45$-24$,-                              ;
                0024          002C          0031        0036      00188          32$-24$,-                                      ;
                                                          001E      00190          31$-24$,-                                      ;
                                                                                   27$-24$,-                                      ;
                                                                                   30$-24$,-                                      ;
                                                                                   29$-24$,-                                      ;
                                                                                   28$-24$,-                                      ;
                                                                                   26$-24$,-                                      ;
                                                                                   25$-24$                                        ;
                                                          0C  BB 00192          PUSHR   #^M<R2,R3>                             ; 0851
                                                          5A  DD 00194          PUSHL   EL_SIZE                                ;
                                     0000V CF             03  FB 00196          CALLS   #3, MOVE_CHAR                          ;
                                                          010A 31 0019B         BRW     45$                                    ;
                                           83             82  7D 0019E 25$:     MOVQ    (BUF_PTR)+, (ELEM_PTR)+                ; 0804
                                                          0104 31 001A1         BRW     45$                                    ; 0799
                                           83             82  D0 001A4 26$:     MOVL    (BUF_PTR)+, (ELEM_PTR)+                ; 0809
                                           83             82  B0 001A7 27$:     MOVW    (BUF_PTR)+, (ELEM_PTR)+                ; 0810
                                                          16  11 001AA          BRB     32$                                    ; 0811
                                           83             82  D0 001AC 28$:     MOVL    (BUF_PTR)+, (ELEM_PTR)+                ; 0816
                                                          0B  11 001AF          BRB     31$                                    ; 0817
                                           83             82  D0 001B1 29$:     MOVL    (BUF_PTR)+, (ELEM_PTR)+                ; 0822
                                                          0C  11 001B4          BRB     32$                                    ; 0823
                                           83             82  D0 001B6 30$:     MOVL    (BUF_PTR)+, (ELEM_PTR)+                ; 0828
                                                       00EC 31 001B9            BRW     45$                                    ; 0799
                                           83             82  B0 001BC 31$:     MOVW    (BUF_PTR)+, (ELEM_PTR)+                ; 0839
                                                       00E6 31 001BF            BRW     45$                                    ; 0799
                                           83             82  90 001C2 32$:     MOVB    (BUF_PTR)+, (ELEM_PTR)+                ; 0844
                                                       00E0 31 001C5            BRW     45$                                    ; 0856
                                           1A             58  D1 001C8 33$:     CMPL    FMT_CODE, #26                          ; 0880
                                                          05  1F 001CB          BLSSU   34$                                    ;
                                           58             19  C2 001CD          SUBL2   #25, FMT_CODE                          ;
                                                          03  11 001D0          BRB     35$                                    ;
                                           58             16  C2 001D2 34$:     SUBL2   #22, FMT_CODE                          ;
                                           09             54  D1 001D5 35$:     CMPL    R4, #9                                 ; 0884
                                                          13  1F 001D8          BLSSU   37$                                    ;
                                                          58  D5 001DA          TSTL    FMT_CODE                               ;
                                                          05  13 001DC          BEQL    36$                                    ;
                                           02             58  D1 001DE          CMPL    FMT_CODE, #2                           ;
                                                          0A  12 001E1          BNEQ    37$                                    ;
                      FF70      CB                        3D  90 001E3 36$:     MOVB    #61, -144(CCB)                         ; 0887
                                50                        04  D0 001E8          MOVL    #4, S                                  ; 0888
                                                          03  11 001EB          BRB     38$                                    ; 0884
                                50                        5A  D0 001ED 37$:     MOVL    EL_SIZE, S                             ; 0891
                                51 00000000'EF48          D0 001F0 38$:         MOVL    AA_IN_CVT[FMT_CODE], R1                ; 0900
                                7E           93           AB  9A 001F8          MOVZBL  -109(CCB), -(SP)                       ;
                                                          50  DD 001FC          PUSHL   S                                      ;
                                                    0C    AC  DD 001FE          PUSHL   ELEM_ADR                               ;
                                                    10    AE  9F 00201          PUSHAB  DSC                                    ;
```

```
                       61          04  FB 00204       CALLS   #4, (R1)                            ;
                                   27  11 00207       BRB     40$                                 ;
        50 00300018    8F          54  78 00209 39$:  ASHL    R4, #3145752, R0                    ; 0937
                                   63  18 00211       BGEQ    43$                                 ;
                       50 00000000'EF44 D0 00213      MOVL    AA_IN_CVT[R4], R0                   ; 0940
                       7E      93  AB  9A 0021B       MOVZBL  -109(CCB), -(SP)                    ; 0942
                       7E      88  AB  98 0021F       CVTBL   -120(CCB), -(SP)                    ; 0941
                       7E      8B  AB  9A 00223       MOVZBL  -117(CCB), -(SP)                    ;
                               DC  AC  DD 00227       PUSHL   ELEM_ADR                            ;
                               14  AE  9F 0022A       PUSHAB  DSC                                 ;
                       60          05  FB 0022D       CALLS   #5, (R0)                            ;
                       75          50  E8 00230 40$:  BLBS    R0, 45$                             ;
                       03      C4  AB  91 00233       CMPB    -60(CCB), #3                        ; 0949
                                   08  13 00237       BEQL    41$                                 ;
            FFFB       8F      C6  AB  B1 00239       CMPW    -58(CCB), #-5                       ; 0950
                                   18  12 0023F       BNEQ    42$                                 ;
                               04  AE  9F 00241 41$:  PUSHAB  DSC                                 ; 0952
                                   01  DD 00244       PUSHL   #1                                  ;
                          0018883C 8F  DD 00246       PUSHL   #1607740                            ;
                       7E      40  8F  9A 0024C       MOVZBL  #64, -(SP)                          ;
          00000000G    00          04  FB 00250       CALLS   #4, FOR$$SIGNAL                     ;
                                   4F  11 00257       BRB     45$                                 ;
     7E        E0      AB          01  C3 00259 42$:  SUBL3   #1, -32(CCB), -(SP)                 ; 0955
                               08  AE  9F 0025E       PUSHAB  DSC                                 ; 0954
                                   02  DD 00261       PUSHL   #2                                  ;
                          00188834 8F  DD 00263       PUSHL   #1607732                            ;
                       7E      40  8F  9A 00269       MOVZBL  #64, -(SP)                          ;
          00000000G    00          05  FB 0026D       CALLS   #5, FOR$$SIGNAL                     ;
                                   32  11 00274       BRB     45$                                 ; 0949
                       7E      93  AB  9A 00276 43$:  MOVZBL  -109(CCB), -(SP)                    ; 0969
                       7E      88  AB  98 0027A       CVTBL   -120(CCB), -(SP)                    ;
                       7E      8B  AB  9A 0027E       MOVZBL  -117(CCB), -(SP)                    ; 0968
                               0C  AE  9F 00282       PUSHAB  F_VALUE                             ;
                               14  AE  9F 00285       PUSHAB  DSC                                 ;
          00000000G    00          05  FB 00288       CALLS   #5, OTS$CVT_T_F                     ;
                       50          5A  D0 0028F       MOVL    EL_SIZE, R0                         ; 0970
                       04          50  D1 00292       CMPL    R0, #4                              ;
                       03          1B  00295          BLEQU   44$                                 ;
                       50          04  D0 00297       MOVL    #4, R0                              ;
                       50          08  C4 0029A 44$:  MULL2   #8, R0                              ;
  0C   BC          50  00          6E  F0 0029D       INSV    F_VALUE, #0, R0, @ELEM_ADR          ;
             FF70  CB          3D  90 002A3           MOVB    #61, -144(CCB)                      ; 0971
                   59          01  D0 002A8 45$:      MOVL    #1, DT_SEEN                         ; 0978
                          FD66 31  002AB              BRW     2$                                  ; 0538
                               04  002AE              RET                                         ; 0985

; Routine Size:  687 bytes,    Routine Base:  _FOR$CODE + 00C1

;  924         0986  1
```

```
;  926     0987  1 ROUTINE DO_READ (                          ! read formatted record and do per-record proc.
;  927     0988  1     FOR$$REC_xn)                            ! adr. or record processing routine
;  928     0989  1     : JSB_DO_READ NOVALUE =
;  929     0990  1 !+
;  930     0991  1 ! FUNCTIONAL DESCRIPTION:
;  931     0992  1 !
;  932     0993  1 !       DO_READ is a local routine which inputs the next record by calling the appropriate
;  933     0994  1 !       record processing routine depending on the statement type
;  934     0995  1 !       (ISB$BSTTM_TYPE) and formal parameter FOR$$REC_xn which
;  935     0996  1 !       is either (1) FOR$$REC_x1 if this is not the last record
;  936     0997  1 !       of the I/o statement or (2) FOR$$REC_x9 if the is the last
;  937     0998  1 !       record of the I/O statement, i.e., this is the end of I/O list call.
;  938     0999  1 !       Then is performs any per-record initialization.
;  939     1000  1 !       Note: DO_READ is called directly from FOR$$UDF_RF9 if
;  940     1001  1 !       next format byte is an end-of-format one, thus saving
;  941     1002  1 !       2 expensive calls to FOR$$UDF_RF1 and FOR$$FMTIN1.  Thus
;  942     1003  1 !       DO_READ has all processing needed to read a record.
;  943     1004  1 !
;  944     1005  1 ! CALLING SEQUENCE:
;  945     1006  1 !
;  946     1007  1 !       JSB DO_READ (R0=for$$rec_xn.s.ar)
;  947     1008  1 !
;  948     1009  1 ! FORMAL PARAMETERS:
;  949     1010  1 !
;  950     1011  1 !       FOR$$REC_xn.s.ar        Adr. of record processing routine (NOT PIC)
;  951     1012  1 !
;  952     1013  1 ! IMPLICIT INPUTS:
;  953     1014  1 !
;  954     1015  1 !       OTS$$A_CUR_LUB          Pointer to current logical unit block
;  955     1016  1 !                               (LUB). Used to setup base pointer ISB
;  956     1017  1 !                               to current I/O statement block
;  957     1018  1 !
;  958     1019  1 ! IMPLICIT OUTPUTS:
;  959     1020  1 !
;  960     1021  1 ! The following locations are set only by previous calls
;  961     1022  1 ! to FOR$$UDF_RF{0,1}, i.e., are effectively OWN for this module.
;  962     1023  1 !
;  963     1024  1 !       LUB$A_BUF_PTR           Pointer: Set to beginning of input record
;  964     1025  1 !       LUB$A_BUF_PTR           Pointer: set to beginning of input record
;  965     1026  1 !       LUB$A_BUF_HIGH          Pointer: set to beginning of input recordn
;  966     1027  1 !       LUB$A_BUF_END           Pointer: set to last char+1 of input record
;  967     1028  1 !--
;  968     1029  1
;  969     1030  2     BEGIN
;  970     1031  2
;  971     1032  2     EXTERNAL REGISTER
;  972     1033  2         CCB : REF $FOR$CCB_DECL;
;  973     1034  2
;  974     1035  2     !+
;  975     1036  2     ! Input record.
;  976     1037  2     ! Return with new beginning and end pointers
;  977     1038  2     ! to next user data buffer to be processed as input.
;  978     1039  2     !-
;  979     1040  2
;  980     1041  2     JSB_REC1 (.FOR$$REC_xn);
;  981     1042  2
;  982     1043  2     !+
```

```
;  983          1044  2     ! Initialize beginning and highest pointer (T format)
;  984          1045  2     ! to the first character position in the input record buffer
;  985          1046  2     !-
;  986          1047  2
;  987          1048  2     CCB [LUB$A_BUF_BEG] = .CCB [LUB$A_BUF_PTR];
;  988          1049  2     CCB [LUB$A_BUF_HIGH] = .CCB [LUB$A_BUF_PTR];
;  989          1050  2     RETURN;                                         ! Return from DO_READ routine
;  990          1051  1     END;                                            ! End of DO_READ routine


                                    60  16 00000 DO_READ:JSB     (FOR$$REC_XN)                    ; 1041
                           BC   AB      B0   AB  D0 00002          MOVL    -80(CCB), -68(CCB)       ; 1048
                           C0   AB      B0   AB  D0 00007          MOVL    -80(CCB), -64(CCB)       ; 1049
                                            05 0000C              RSB                              ; 1051

; Routine Size:  13 bytes,    Routine Base:  _FOR$CODE + 0370

;  991          1052  1
```

```
;  993          1053  1 GLOBAL ROUTINE FOR$$UDF_RF9                    ! Formatted input - end of I/O list call
;  994          1054  1     : JSB_UDF9 NOVALUE =
;  995          1055  1
;  996          1056  1 !++
;  997          1057  1 ! FUNCTIONAL DESCRIPTION:
;  998          1058  1 !
;  999          1059  1 !       FOR$$UDF_RF9 performs end of I/O list input formatting.
; 1000          1060  1 !       It only calls the FOR$$UDF_RF1 if there were no I/O list
; 1001          1061  1 !       elements at all, else it need do nothing.
; 1002          1062  1 !
; 1003          1063  1 !       All format codes are processed until a data transmitting
; 1004          1064  1 !       format code is encountered (or colon) or end of format.
; 1005          1065  1 !
; 1006          1066  1 ! CALLING SEQUENCE:
; 1007          1067  1 !
; 1008          1068  1 !       JSB FOR$$UDF_RF9 ()
; 1009          1069  1 !
; 1010          1070  1 ! FORMAL PARAMETERS:
; 1011          1071  1 !
; 1012          1072  1 !       NONE
; 1013          1073  1 !
; 1014          1074  1 ! IMPLICIT INPUTS:
; 1015          1075  1 !
; 1016          1076  1 !       See FOR$$UDF_RF1
; 1017          1077  1 !
; 1018          1078  1 !
; 1019          1079  1 ! IMPLICIT OUTPUTS:
; 1020          1080  1 !
; 1021          1081  1 !       See FOR$$UDF_RF1
; 1022          1082  1 !
; 1023          1083  1 ! FUNCTION VALUE:
; 1024          1084  1 !
; 1025          1085  1 !       NONE
; 1026          1086  1 !
; 1027          1087  1 ! SIDE EFFECTS:
; 1028          1088  1 !
; 1029          1089  1 !       See FOR$$UDF_RF1
; 1030          1090  1 !--
; 1031          1091  1
; 1032          1092  2     BEGIN
; 1033          1093  2
; 1034          1094  2     EXTERNAL REGISTER
; 1035          1095  2         CCB : REF $FOR$CCB_DECL;
; 1036          1096  2
; 1037          1097  2     !+
; 1038          1098  2     ! If there were no items in I/O list, then the current format
; 1039          1099  2     ! character is zero.  In this case, call FOR$$UDF_RF1 to execute
; 1040          1100  2     ! non data-transmitting format codes.  Otherwise, do nothing
; 1041          1101  2     ! because we have already executed all required formats.
; 1042          1102  2     !-
; 1043          1103  2
; 1044          1104  2     IF .CCB [ISB$B_FMT_CODE] EQL 0 THEN FOR$$UDF_RF1 (0, 0, 0);
; 1045          1105  2
; 1046          1106  2     RETURN;
; 1047          1107  1     END;                                          ! End of FOR$$UDF_RF9 Routine
```

```
                                 BF    AB  95 00000 FOR$$UDF_RF9::
                                                     TSTB    -113(CCB)                ; 1104
                                       09  12 00003         BNEQ    1$
                                       7E  7C 00005         CLRQ    -(SP)
                                       7E  D4 00007         CLRL    -(SP)
                               FD36    CF  03  FB 00009     CALLS   #3, FOR$$UDF_RF1
                                           05 0000E 1$:     RSB                      ; 1107

; Routine Size:  15 bytes,    Routine Base:  _FOR$CODE + 037D

; 1048          1108  1
```

```
;  1050          1109  1 ROUTINE MOVE_CHAR (                          ! Move characters
;  1051          1110  1     LEN,                                     ! Fill length
;  1052          1111  1     SOURCE,                                  ! Source address
;  1053          1112  1     DEST)                                    ! Destination address
;  1054          1113  1     : NOVALUE =
;  1055          1114  1
;  1056          1115  1 !++
;  1057          1116  1 ! FUNCTIONAL DESCRIPTION:
;  1058          1117  1 !
;  1059          1118  1 !       MOVE_CHAR moves characters from one string to another.  It is
;  1060          1119  1 !       identical to CH$MOVE except that it does not return a value.
;  1061          1120  1 !       A separate called routine is used so that registers R0 through
;  1062          1121  1 !       R5 are free in the calling routine.
;  1063          1122  1 !
;  1064          1123  1 ! CALLING SEQUENCE:
;  1065          1124  1 !
;  1066          1125  1 !       CALL MOVE_CHAR (len.rwu.v, source.rbu.r, dest.wbu.r)
;  1067          1126  1 !
;  1068          1127  1 ! FORMAL PARAMETERS:
;  1069          1128  1 !
;  1070          1129  1 !       len             Number of bytes to move.
;  1071          1130  1 !       source          Address of string to move from.
;  1072          1131  1 !       dest            Address of string to move to.
;  1073          1132  1 !
;  1074          1133  1 ! IMPLICIT INPUTS:
;  1075          1134  1 !
;  1076          1135  1 !       NONE
;  1077          1136  1 !
;  1078          1137  1 ! IMPLICIT OUTPUTS:
;  1079          1138  1 !
;  1080          1139  1 !       NONE
;  1081          1140  1 !
;  1082          1141  1 ! FUNCTION VALUE:
;  1083          1142  1 !
;  1084          1143  1 !       NONE
;  1085          1144  1 !
;  1086          1145  1 ! SIDE EFFECTS:
;  1087          1146  1 !
;  1088          1147  1 !       NONE
;  1089          1148  1 !
;  1090          1149  1
;  1091          1150  1 !--
;  1092          1151  2     BEGIN
;  1093          1152  2     CH$MOVE (.LEN, .SOURCE, .DEST);
;  1094          1153  1     END;


                                        003C 00000 MOVE_CHAR:
                                                   .WORD   Save R2,R3,R4,R5                ; 1109
                    0C   BC       08   BC       04   AC  28 00002         MOVC3   LEN, @SOURCE, @DEST     ; 1152
                                                   04 00009         RET                     ; 1153

; Routine Size:  10 bytes,    Routine Base:  _FOR$CODE + 038C
```

```
ROUTINE COPY_CHAR (                                          ! Copy characters
    SOURCE_LEN,                                              ! Length of source
    SOURCE_ADDR,                                             ! Address of source
    DEST_LEN,                                                ! Length of destination
    DEST_ADDR)                                               ! Address of destination
    =

!++
! FUNCTIONAL DESCRIPTION:
!
!       COPY_CHAR moves characters from one string to another, blank padding
!       if necessary.  It is equivalent to a CH$COPY with a blank fill.
!       A separate called routine is used so that registers R0 through
!       R5 are free in the calling routine.
!
! CALLING SEQUENCE:
!
!       pointer.rbu.v =  COPY_CHAR (source_len.rwu.v, source_addr.rbu.r, dest_len.rwu.v, dest_addr.wbu.r)
!
! FORMAL PARAMETERS:
!
!       source_len      Number of bytes in source
!       source_addr     Address of source
!       dest_len        Number of bytes in destination
!       dest_addr       Address of destination
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! FUNCTION VALUE:
!
!       The address of the next byte past the destination.
!
! SIDE EFFECTS:
!
!       NONE
!
!--

  BEGIN
  RETURN CH$COPY (.SOURCE_LEN, .SOURCE_ADDR, %C' ', .DEST_LEN, .DEST_ADDR);
  END;


                                         003C 00000 COPY_CHAR:
                                                          .WORD   Save R2,R3,R4,R5                                  ; 1154
 0C   AC        20     08  BC    04  AC  2C 00002         MOVC5   SOURCE_LEN, @SOURCE_ADDR, #32, DEST_LEN, -         ; 1199
                                 10  BC     0000A                 @DEST_ADDR                                        ;
                       50        53  D0     0000C         MOVL    R3, R0                                            ;
```

```
                                        04 0000F        RET                                             ; 1200

; Routine Size:  16 bytes,    Routine Base:  _FOR$CODE + 0396

; 1143          1201  1 END                                   ! End of FOR$$UDF_RF Module
; 1144          1202  1
; 1145          1203  0 ELUDOM



;                                  PSECT SUMMARY

;       Name                    Bytes                   Attributes

;  _FOR$CODE                     934  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _FOR$DATA                     120  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)



;                               Library Statistics

;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1     9776        12         0         581       00:01.0
;  _$255$DUA28:[FORRTL.OBJ]FORLIB.L32;1   711       209        29          52       00:00.6
;  _$255$DUA28:[FORRTL.OBJ]RTLLIB.L32;1    36         0         0           8       00:00.1



;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:FORUDFRF/OBJ=OBJ$:FORUDFRF MSRC$:FORUDFRF/UPDATE=(ENH$:FORUDFRF)

; Size:         880 code + 174 data bytes
; Run Time:       00:25.1
; Elapsed Time:   00:59.3
; Lines/CPU Min:   2872
; Lexemes/CPU-Min: 17777
; Memory Used:  308 pages
; Compilation Complete
```

FORSTOP
LIS

FORUDFR
LIS

FORTIMEDS
LIS

FORUDFWN
LIS

FORUDFW
LIS

FORUDFRN
LIS

FORUDFRU
LIS

FORTIME
LIS

FORUDFWF
LIS

FORUDFRF
LIS